The Advantages and Enforcement of Open Source Software in Embedded Real-time Systems

G.Thanos*, C.Kavadias*, A.Litke*, G.J.Wintterle**

*TELETEL S.A; 124, Kifisias Avenue, Athens, GREECE
Tel: + 30 10 6983393; Fax: +30 10 6983391;
**ALCATEL SEL; 10, Lorenzstr, Stuttgart, GERMANY
Tel: + 49 711 821 46443; Fax: +49 711 821 42578;

E-mail: G.Thanos@teletel.gr

Abstract: Work in this paper identifies the characteristics of open source software and outlines the perceived advantages of using open source software in embedded real-time system. Within this context, the LIBRE IST Project is presented, explaining its objectives and contribution in the real-time embedded applications market.

Keywords: Open source software, LIBRE, Linux.

1. Open Source Software Initiative

The main features that characterise free (open source) software is the freedom that users have to:

- Use the software as they wish, for whatever they wish, on as many computers as they wish, in any technically appropriate situation.
- Have the software at their disposal to fit it to their needs. Of course, this includes improving it, fixing its bugs, augmenting its functionality, and studying its operation.
- Redistribute the software to other users, who could themselves use it according to their own needs. This redistribution can be done for free, or at a charge, not fixed beforehand.
- Users of a piece of software must have access to its source code.

The source code of a program, usually written in a high level programming language, is absolutely necessary to be able to understand its functionality, to modify it and to improve it. If programmers have access to the source code of a program, they can study it, get knowledge of all its details, and work with it as the original author would.

Paradoxically, if this freedom is to be guaranteed for a given piece of software, with current legislation, it is necessary to “protect” it with a licence which impose certain restrictions on the way that it can be used and distributed (as it will be shown later). This fact causes some controversy in certain circles, because it is considered that these
licences make the software distributed under them “less free”. Another view, more pragmatic, is that software will be made more free by guaranteeing the perpetuation of these freedoms for all its users. Because of that, people holding this view maintain that it is necessary to limit the ways of use and distribution. Depending on the ideas and goals of the authors of a piece of code, they can decide to protect it with several different licences.

2. Advantages of Open Source Software

Motivations for using and developing open source software are mixed, ranging from philosophical and ethical reasons to pure practical issues. In this subsection, some of the most widely proposed practical advantages will be introduced.

Usually, the first perceived advantage of open source models is the fact that open source software is made available gratis or at a low cost. But this characteristic is not exclusive to open source software, and several proprietary software products are made available in similar ways (a prominent case could be Microsoft’s Internet Explorer). What really distinguishes open source software from software available without fee is the combination of effects due to the characteristics discussed previously. All of them combined produce a synergistic impact, which is the cause of the real advantages of the open source model. Let us provide some more detail on how do these characteristics turn into advantages:

The availability of the source code and the right to modify it is very important. It enables the unlimited tuning and improvement of a software product. It also makes it possible to port the code to new hardware, to adapt it to changing conditions, and to reach a detailed understanding of how the system works. This is why many experts are reaching the conclusion that to really extend the lifetime of an application, it must be available in source form. In fact, no binary-only application more than 10 years old now survives in unmodified form, while several open source software systems from the 1980s are still in widespread use (although in many cases conveniently adapted to new environments). Source code availability also makes it much easier to isolate bugs, and (for a programmer) to fix them.

The right to redistribute modifications and improvements to the code, and to reuse other open source code, permits all the advantages due to the modifiability of the software to be shared by large communities. This is usually the point that differentiates open source software licences from “nearly free” ones. In substance, the fact that redistribution rights cannot be revoked, and that they are universal, is what attracts a substantial crowd of developers to work around open source software projects.

The right to use the software in any way. This, combined with redistribution rights, ensures (if the software is useful enough), a large population of users, which helps in turn to build up a market for support and customization of the software, which can only attract more and more developers to work in the project. This in turn helps to improve the quality of the product, and to improve its functionality. Which, once more, will cause more and more users to give the product a try, and probably to use it regularly.
The issue about non-exclusive rights on the software, which has just being mentioned, deserves some more attention. When no one holds exclusive rights on a given code (sometimes mentioned as “life or death rights”), several traditional problems of the proprietary software model can be overcome:

*There is no one with the power to restrict in a unilateral way that the software is used,* even in a retroactive way. Such a power manifests, for instance, when a proprietary software vendor decides not to upgrade some software product for some old platform. In this case, customers can only stick to the old version of the software, or switch to another product. If open source software is used, customers can also fund some development for the desired platform, or look for other vendors to provide the upgrades (of the very same product).

*There is no single entity on which the future of the software depends.* This is a very common concern with proprietary software. Let us say that a company uses a software product, and relies on the software manufacturer for upgrades and continued development. If the software manufacturer closes doors, or decides to discontinue development of the product, no one has the right to take the program and continue development on it, effectively killing its usability in the market. This has happened many times, and this problem is amplified by the recent mergers in the software market, that usually lead to “cannibalization” of some software product to allow just one or two to get to the market. Open source software effectively protects against this, because if the group or company that originated the code decides to stop development, it is always possible to fund another software group to continue the maintenance and improvement, without legal nor practical limitations.

*No “black boxes” are possible.* This point is so important that open source is now considered by many experts as one of the necessary conditions for dependable applications. There are several reasons for this importance. One of them is related to the dependability of the services provided by a given software. By having the source code available, it is possible to perform a thorough inspection and verify the correctness of the algorithm and the implementation scheme used. This is also possible in part even with closed source or nearly free licences. The difference lies in the fact that users are allowed to modify everything they find appropriate to suit their needs. A glaring example is the Linux kernel and its “international patches”, a set of enhancements with legal problems in some countries. These patches include support for encrypted communications, and can be legally used in large parts of the world. The patches have been developed by concerned parties in countries where such a development is allowed, and therefore users in those countries can use those enhancements. With binary only products no inspection is possible, with closed source or nearly free licences inspection is possible, but modifications are forbidden, so the inherent advantage of having source code available is rendered ineffective.

*There is always the possibility of “forking”,* or creating an alternative code base if the current one is in some way perceived as wrongly managed. This is sometimes considered a disadvantage, having to manage not only one code base, but two. A “fork” is a subdivision of the code base in two different parts, each managed by a different group. Forks happens for technical or licence reasons, for example because a particular release is made under a non-free licence, the previous one is used as a base
for subsequent free releases. Technical motivations are common, because there are sometimes many different ways to perform a task, and it is not possible to decide which is better. So if the two camps cannot reach a consensus and the user base is large enough the code splits in two, and both continue development. If the reasons for the split are overcome, usually the two camps agree on a reunification. An example is the Linux kernel, where two distinct code bases are used, one “stable” and one “experimental”. This way it is possible to introduce new and potentially dangerous technologies without disrupting the stable ones. So, people interested in leading edge technologies can try them, and people that use the Linux kernel in production environments can count on stable and tested features.

*No per-copy fees can be asked for modified versions*, and anyone can use the current code base to start new projects. Working knowledge can be gathered at a minimal cost. This is what made Internet software systems such an important factor in the new economy: students, and people trying new technologies were able to integrate and adopt them immediately, without the hurdles of commercial or non-disclosure licence agreements. In addition, the right to freely modify them allowed for the incredible expansion in the number of communication protocols and systems, each perfectly tailored to the needs of their users. This is also a reason for the overwhelming success of the Linux kernel, widely employed by students thanks to its near-zero cost, and subsequently used by the same students in the startups originated by them, when they turn into entrepreneurs after leaving University.

*There are fewer conflicting priorities due to marketing pressures.* This is a simple consequence of the fact that there is no single commercial entity pushing for precise delivery dates or features that must be supported. Usually open source software is delivered “when it is ready”, and when the development team feels that its quality is good enough. This means that software usually does not need as many “service packs”, updates and such, reducing the maintenance cost. Of course this could be turned into disadvantage if a product is indefinitely delayed, or if some feature is missing one release after another. But in this case, the competition between projects may help. If a project starts failing to meet the expectations of its users, it often happens that a new project is forked, using the same code base, to fill this gap. This happens especially if a market exists for some new features, or for better quality versions of the application.

### 3. The LIBRE Experiment

The work presented in this document is performed in the context of the LIBRE (Advanced LINUX-Based Products for Embedded Telecommunication Solutions) IST-2000-29313 project. ALCATEL SEL and TELETEL SA form the LIBRE consortium.

There is a clear demand from both partners of the LIBRE project to gain advantage in embedded software applications by having full access to every piece of software. This is especially true for embedded telecoms applications where timing aspects and full control over them is of highest importance. A typical example of a low level telecommunications protocol layer the MTP-2 protocol for SS7 networks. Whilst timers for connection establishment in the network layer may be in the range of 4-20
seconds, the MTP-2 timer T5, for example, defines a response time of 80-120 ms. It is apparent from the above that the full control over any underlying software allows the LIBRE partners to improve critical points and to provide the necessary functionality for their applications.

On the other hand LINUX is the most well know open source OS, supported by a huge number of developers worldwide being so far a real success story in the area of open source software. From a purely technical point of view, LINUX shares many of the strengths of Unix where it has its roots (industry standard application programming interfaces (APIs), a rich feature set robustness and stability through its use of memory protection, fully integrated networking and a wealth of application developers). In addition, the open source movement has allowed LINUX to avoid the diversification that was the bane of Unix. More recently, the real-time, embedded systems community has started become more and more interested in LINUX and now there are already various version of embedded LINUX available. The additional strengths of the Open Source LINUX is the proliferation of Open Source application software available in the LINUX community.

LINUX first appeared as a general purpose OS. It is designed to optimise overage performance and try to give to each process a fair share of computing time. Being a general purpose OS creates some problems when ported to embedded systems such as.

- It has a non-preemptive kernel.
- Low Real-Time Performance (non-preemptive kernel, disables interrupts for long periods of time)
- Reduced Scalability
- Lack of Real-Time APIs
- Has a kernel that disables interrupts for long periods of time

However, the LIBRE project has the challenge to make use of the sophisticated services and highly optimised average case behaviour of a standard time-shared computer system while still identifying and permitting real-time functions to operate in a predictable and low-latency environment.

The additional strengths of the Open Source LINUX is the proliferation of Open Source application software available in the LINUX community. Especially as far as it concerns networking, a large number of protocol stacks are available, such as:

- IP Stack: TCP/UDP IP, ICMP, SMTP, SNMP etc.
- ATM: UNI 3.0/UNI 3.1/UNI 4.0 unicast signalling, AAL0, AAL5, IP over ATM, LAN Emulation

The main objective of the LIBRE project is to integrate and evaluate the open source embedded LINUX Operating System (OS) in the development lifecycle of Embedded Software Products. The project addresses the area of embedded telecommunication applications and will perform a Best Practice Experiment in the scope of a broader project in order to assess the suitable use of LINUX in terms of development effort, reusability, reliability and performance.
The LIBRE project will perform a specific experiment, which is a part of a larger industrial project, namely the development of an advanced interworking unit between various networks for the migration to 3rd generation mobile communications (Figure 1).

![Figure 1: The LIBRE Application Experiment](image)

### 4. Conclusions

The LIBRE objectives and anticipated results well represent the needs of European Industrial Telecommunication Sector acting in a competitive and international market. Currently, there are many different LINUX distributions by a number of commercial and not commercial organisations. Each distribution adds or enhances the kernel of the operating system and usually comes completely pre-configured to specifications set by its organisation. LINUX is now a proven solution as a reliable, efficient, stable and high-performance OS for a variety of different systems and applications, being used by many manufacturers and individual users all over the world. Utilising LINUX in real-time embedded systems where requirements like reliability and stability become critical will certainly solve a lot of problems that systems of this kind suffer and will form a very strong open software tool in the hands of the developers.